A Survey on Rule-Based Systems and the significance of Fault Tolerance for High-Performance Computing

G. Sreenivasulu
Department of CSE, J B Institute of Engineering and Technology, Hyderabad, India
svgonuguntla@gmail.com

P. V. S. Srinivas
Sreenidhi Institute of Technology and Science, Hyderabad, India.
pvssrinivas23@gmail.com

Dr. A Govardhan
Department of CSE, Jawaharlal Nehru Technological University Hyderabad, India
govardhan_cse@yahoo.co.in

***Abstract*— The high-performance computing (HPC) systems participate an significant responsibility in many highly computational applications and systems. Understanding the failure behavior of such a massively parallel system is essential to accomplishing high utilization of large systems. This process requires continuous on-line monitoring and analysis of all incidents generated in the system, including long-term normal notification, performance metrics, and failures. This article illustrates the significance of HPC-Ss (HPC-S) and their fault tolerance, especially rules-based systems. To explore the efficient Fault-Tolerant (FT) mechanism and fault prediction method for an efficient FTmechanism in distributed systems with different rules. We also analyzed the progress of HPC in the rule-based distributed system and its future development direction.**

Keywords- HPC, Rule-Based, Fault tolerance, Distributed Systems

I. INTRODUCTION

Due to massively parallelization, the complication of computer architectures has made the process of developing High-performance computing (HPC) systems very challenging. The HPC systems (HPC-S) participate an important responsibility in today's society. Widespread applications consist of "weather forecasting", "aircraft collision simulation", "computational fluid dynamics in aerodynamic studies", "bioinformatics", "molecular modeling of protein folding in biomedical research", etc. [2], [3], [4], [5]. However, many of the procedural confronts that HPC-S face as they produce to higher levels because of the amplified numeral of processors complicated, due to the indicate instance among failures of individuality structure sections and the represent time between malfunction of the entire system [7], [9], [10]. As programs and information spread transversely "hundreds of thousands of different processors" and "separate memory libraries", the organization of data exchange and sequential computation necessitates extremely composite reason, a large quantity of dedicated training and fault tolerance. The majority programs depends on a messaging documentation that necessitates moderately short logic and commands. It may require concentrated debugging and optimization tools production.

Cloud Computing delivers innovative computational concepts, capability, and agility to "HPC applications" by providing numerous "virtual machines (VMs)" for compute-demanding applications that use cloud services. It is likely that compute-demanding applications determination be installed more and more in HPC-S in cloud computing [15], [16]. For instance, an "Amazon Elastic Compute Cloud (Amazon EC2)" [17] cluster.

The key challenges facing HPC-S mentioned in the literature can be separated into (1) fault tolerance and (2) rollback recovery. These are two major issues facing the HPC-S. The reason of this learning is to improve the fault tolerance of HPC-S through rule-based prediction and a new rollback recovery scheme. The current research is studied in detail.

A. Fault Tolerance

Current research by Schroders and Gibson [8], Eggetoha et al. [11] and Yigitbasi, et al. [12] and the data collections make available in [25] demonstrate that HW such as "processor, hard drive, integrated circuit socket, and memory", caused additional 50% of malfunction on HPC-S. These works moreover demonstrate:

- The malfunction rate is approximately comparative to the many CPUs.
- The strength of the efforts affects the failure rate [8].
- There is a association with the malfunction rate over time [12], [13].

B. Rollback-Recovery

The "Rollback Recovery Fault Tolerance (RR-FT) techniques" are frequently utilized by the HPC-S community [14], [19]. When individual or more computational nodes fail, they tend to reduce the impact of the malfunction on "compute-intensive applications" that execute on HPC-S. A high-quality illustration of RR-FT is "checkpoint" and "restart". Checkpoints and restarting allow computationally demanding difficulties to facilitate to acquire extended period to accomplish on HPC-S to restart commencing a checkpoint in the occurrence of an error or failure.

Nevertheless, current publications [1], [2], [4], [10], [16] show that as the numerous elements in nowadays HPC-S are prolongs to develop and the applications operation on HPC-S possibly will not be capable to communicate with fundamental

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

checkpoints and restart the method to progress. It is for the reason that the system will utilize mainly of the instance on the checkpoint, which is not element of the calculation activity. It is especially important that FT solutions will diminish the overhead of rollback recovery [35].

The following article reviews the significance of HPC in the second part. The FT mechanism in HPC-S is discussed in Section 3, and the rule-based HPC distributed system is discussed in Section 4. In the 5 and 6 sections, the related work was investigated and the conclusion of the evaluation was summarized.

C. The exploit of RTT to Infer Congestion

However, the Internet needs to provide some outline of feedback to data traffic originating from the congested links so that it can regulate its transmission rate depending on the accessible bandwidth, effectively managing end-to-end congestion control [16]. Feedback on congestion can be implicit or explicit. In the case of implicit response, the network's transport layer protocol attempts to maintain high throughput by approximation "service time", "end-to-end delay", and "packet deliver fail". The TCP protocol widely used by the Internet [7], [10] implicitly feeds back lost packets over time and repeatedly. Terminal nodes usually deploy explicit feedback. However, relying on end nodes for implicit or explicit feedback is not enough to achieve a high throughput of the Internet.

II. Significance Of HPC

The HPC-S dates in past to the 1960s whilst it utilized "parallel and distributed computing" to accomplish high computational concert. The "Parallel computing" utilizes contribute to memory to substitute an information among processors, whereas "distributed computing" utilizes distributed memory to share information among the processors through messaging. In recent times, as parallel computers tend to have some distributed features, it is difficult to distinguish among the parallel and distributed systems.

Currently, "parallel and distributed computing systems" through a huge amount of processors be often referred to as HPC-S. The HPC-S extent from hundreds of processors to hundreds of thousands of processors, for instance the "Clay Titan" [22]. For a moments referred to as "long-executing applications" and "compute-intensive applications" be frequently "scientific calculations" that analyze and solve large and complex scientific problems using "mathematical models" and "quantitative analysis techniques" executing on HPC-S. With cloud computing, HPC-S are no longer confined to huge organizations but moreover to characters. The "Cloud computing" [15] has many advantages, together with no up-front speculation in the procure and setting up of tools and software. The HPC-S in the cloud is a high-quality substitute for conventional HPC-S.

The FT is a system usually computer-based property that continues to operate normally in the occurrence of some component failures [23]. The HPC-S really wants FT because it ensures that "compute-intensive applications" are concluded in time. In various FT systems, a mixture of one or additional techniques is utilized.

III. Fault Tolerance Mechanisms In HPC Systems

The HPC-S FT is main key confronts facing HPC application in cloud services. There is evidence that a system with "100,000 processors" experiences a processor breakdown each hours [18]. A breakdown take places whilst a HW element fails and requires to be replaced when a "software component fails", "node/processor is stop", or "force a restart", or the software can not absolute a execution. In this matter, the application that uses the unsuccessful component will not succeed.

In accumulation, HPC applications implemented in the cloud execute on "virtual machines", which are further probable not succeed because of "resource sharing" and "contention" [2], [8], [12], [24], [27]. As a result, "FT technology" is especially significant for "HPC applications" that operate in a cloud environment since FT indicates that operating costs and resource consumption are reduced by eliminating the need to restart long-executing applications from scratch in the occurrence of a malfunction.

The availability of interconnected networks in HPC-S is the foundation for the continued execution of large-scale applications. In the incident of a failure, the interconnect recovery mechanism coordinates complex operations to restore the network connection between nodes. As the size and design complexity of HPC-S increase, the system is susceptible to failure during the performance of the interconnect recovery process. HPC components continue to grow to achieve higher performance and meet the requirement of science and other application users. To reduce the average repair time and increase the availability of these systems, a FT solution is required.

The significance of HPC fault-tolerant systems has been extensively recognized by a variety of research institutions in HPC-S. Various schemes [1], [3], [8], [9], [19] have been proposed to offer FT in HPC-S. In this approach [1], [2], [14] explore redundancy and rollback recovery techniques, respectively. Rollback Recovery [1], [14] is one of the most extensively utilized FT mechanisms in HPC-S. Rollback recovery includes checkpoints, fault detection and recovery/restart [6], [8]. Nevertheless, "rollback recovery" typically enlarges the completing period of HPC applications, growing the resource usage and costs of executing HPC applications in legacy HPC-S and in current distributed HPC-S such as the cloud.

A. Software Failure

It is significant to point out that in most systems, 20-30% of the failures are root cause uncertain. Since the percentage of HW breakdowns is greater than the proportion of undecided breakdowns in all systems and the proportion of software malfunctions is nearly the proportion of undecided breakdowns, we be able to still finish that HW & SW are one of the biggest suppliers to failure. Nevertheless, it cannot conclude that several other source of crash besides "human", "environment", "network" is negligible.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

In all-purpose, HW issues are intimately observed and properly supervised through the administration system. In adding up, the HW is simple to analyse than software. The distribution of the number of nodes involved in the failure caused by different types of root causes is also different. Faults with HW root cause propagate to a minimum of node aggregates outside the bounds of a small proportion. In contrast, software failures propagate in large proportions. And, even for the same HW problems, different times can be fixed depending on when it happens. For example, "CPUs", "memory", and "node interconnect" difficulties represented by the variation coefficient of LANL systems have a variability of repair times of 36, 87, and 154, respectively. This shows that convenient are erstwhile features that can lead to a high degree of variability in HW breakdowns. Software breakdowns have comparable behavior.

Likewise, El-Sayed and Schroeder [28] considered the field malfunction data for high-performance concrete systems available in [29]. The failure data study was collected for nine years. They observed a considerable association among the network, environment, and software breakdowns. They moreover examined that convenient was a significant increase in the likelihood of software breakdown after a influence problem happened.

Nagappan et al. [30] of "North Carolina State University" studied on-demand practical computing lab failure log files. The "North Carolina State University's virtual computing lab" runs as a confidential cloud through additional than "2000 computers". In this learning, they examined to facilitate system software played a comparatively negligible function in system failures. According to their conclusion, the majority of documented breakdowns were reasoned through "workload", "license depletion", and "HW failures".

B. Hardware Failure

The "USENIX Association" [24] also released a large number of breakdown data, "Computer Failure Database (CFDR)". It includes failure statistics for "22 HPC-S", together with a entirety of "4,750 nodes" and "24,101 processors" accumulated by "Los Alamos National Laboratory (LANL)" over a nine-year period. Workloads include "3D scientific simulations" of great-scale, long-executing operations that acquire months to comprehensive.

To additional check the breakdown rate, we particular seven HPC-S commencing the breakdown database [25]. The system of choice includes the largest total numeral of CPUs and/or clusters of five compute nodes, a "symmetric multiprocessing (SMP) system" through the utmost number of CPUs, and the only "non-uniform memory access (NUMA)" system in the data deposits.

Schroeder and Gibson [8] explored fault data accumulated at two huge HPC sites: "datasets from LANL RAS" [25], and over a year on a huge "supercomputing system" by means of "20 nodes" and over "10,000 processors" data accumulated throughout the period. Their analysis shows that:

- The average repair time for all faults (regardless of fault type) is regarding 6 hours.
- There is a association among the breakdown rate of the system and the functions it executing.
- Some systems may fail three times in 24 hours.
- The breakdown rate is approximately relative to the amount of processors in the system.

C. Human Error Failure

Oppenheimer and Patterson [33] reported that operative error was one of the single leading reasons of breakdown. According to their description, "Architecture and Dependability of Large-Scale Internet Services", the operational staff have replaced systems such as HW replacement, system reconfiguration, deployment, patches, software upgrades, system maintenance and other failures. They attribute "14-30% of failures" to human inaccuracy.

It follows that approximately every breakdowns in "compute-intensive applications" are caused by "HW failures", "software failures", and "human error". Nevertheless, the only foremost reason why failure is clear is difficult, as the analysis reported above is ongoing:

- Utilize different systems executing different applications;
- Beneath diverse background circumstances, and
- Use dissimilar data correlation phases and process.
- Therefore, effectual fault-tolerant HPC-S have to deal with the HW & SW malfunctions as healthy as human mistakes.

IV. Rule-Based Distributed Systems For HPC

The rule-based expert system's early relative success with a more efficient rule-based reasoning engine has driven the application of rule technology to "distributed computing" and "multi-agent systems". This investigate trend follows the parallel promotion of "rule-based reasoning", as fine as the "distributed computing model". There are at slightest of two inclinations that able to be experiential at this time: (i) improved "inference algorithms" for rules systems by means of "parallel and distributed system" technologies; (ii) the additional descriptive environment of "rule-based languages" than procedural languages towards develop a more complex system of autonomous components called "software agents".

A. Parallel Computing for Rule-based Systems

The evolution of computational representations for "rule-based production systems" is principally correlated to the development of the "RETE algorithm" [6] and its extension to the well-organized identical of regular outlines and operational memory constituents [4], except to the simultaneous processing of rules and the establishment of creation systems From the past half of the 1980s, a vivid research route started in the 1990s. The main result of these studies is the powerful implementation of rules and systems for the development of technology.

The researchers of [16] suggested a innovative parallel architecture for nested parallelism of "forward-link inference algorithms" that take advantage of "rule-based production

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

systems" on multi-processor systems. The foremost result of this effort was a marked enhancement in the speediness with which "rule-based production systems" were implemented, articulated as rule cycles / second, and changes in functioning memory of elements per second. Their process deal with the entire stages of the familiar link reasoning cycle through: "matching", "parsing", and "right-hand rule evaluation".

The researchers of [20] proposed an thoroughly investigation of concurrency calculation methods to advance the development of one or more regulation schemes. The author first considers several of the development of "rule-based systems" and later discuss the following as, (i) "parallel production systems, algorithms" that include parallel rules loops, (ii) "distributed production systems" beneath "distributed control", and "multi-agent production systems" and their associated managing concerns.

The researchers of [21] recommend a "parallel and distributed" versions of the "RETE algorithm" using "master-slave paradigms". The outline corresponding system breaks down into master and slave components to work in comparable. Every element contains a duplicate of the "RETE network". The convention are stimulated in similar by the main modules. As soon as a regulation is formulate active, it transmits the entire activated evidence to an existing needy element for commencement and arrivals the result. Consequently, the regulations able to be triggered in parallel with the active calculation distributed between the dependent modules.

B. Agent Reasoning Models for Rule-Based Systems

Earlier work suggested the utilize of "rules-based systems" as the essential interpretation model for agents acting as fraction of a "multi-agent system". With this schemes, every agent in the system consist of a rules mechanism, so its activities can decreased to enforcing "rules-based reasoning". The agent synchronization be able to accomplished by sharing a "functioning memory" or "asynchronous messaging".

In [25] a "multi-agent system" known as "MAGSY" is described, where every agent is a "rules-based system". Agents can correspond asynchronously and serve other agents. The "MAGSY" is actually a generic multi-agent framework. Each "MAGSY agent" is a harmony of evidence, regulations, and services. Agents capable of receive information from other agents to trigger an keep posted of their facts. Agents able to call a services offered through last agents. As a consequence, service implementation able to transform the agent's evidence and regulations. Every "MAGSY agent" achieves "rule-based reasoning" with a "forward chain rule interpreter" depend on the familiar "RETE algorithm".

Researchers [26] and [27] believe "multi-agent production systems", which differ theoretically since "parallel and distributed production systems". Although concurrent creation systems give emphasis to "parallel rules matching", and "ring and distributed production systems" highlight the active distribution of production amongst the main groups in an association and aim at improving "execution performance", "multi-agent production systems" focus on multiple self-determining production systems acting together Working memory, valuable for their management.

C. HPC for Rule-based Systems

Recent research developments can be experiential when studying the synergies between HPC and rules-based systems and reasoning. In case of, the senior expression of "rule-based language" determines the calculation difficulty of presumption algorithms, thus preventive the probable of "rule-based systems" in applications that necessitate big-extent inference. The availability of HPC opens up innovative potential for "scalable rule reasoning" in distributed systems. HPC-S consist of "supercomputers", "computer clusters", "grids" and, more recently, "cloud computing infrastructures".

It may be the first report [31] that uses the results of parallelization of the deductive database and the accessibility of clusters in parallel to consider the enhancement in "semantic web reasoning". The researchers of [32] recommended a data partitioning method, a "parallel algorithm", and numerous optimizations to the "scalable parallel reasoning" of "materialized OWL repositories". The functioning of this algorithm is supported on "Jena open-source rule reasoning" and experiments on a "16-node computer cluster".

In "MARVIN", a parallel and distributed stage for processing great number of "RDF data", is described in [33] and uses a new policy identified as "split-exchange on loosely coupled peer-to-peer network". The initiative of this scheme is to split the "RDF triples" continually, calculate the closures of every separation in parallel, and after that exchange the separations through switch over among peers. This method proved to ultimately accomplish the integrity of interpretation and proposed an effective scheme called "SpeedDate" for substitute data among the peers.

The "Map-Reduce" is a procedure for writing large-scale data computing responsibilities on big "computer clusters" [34]. The "Hadoop" is an "Apache project" that build ups "open-source software" for reliable, scalable, distributed computing" and provides the "Map-Reduce programming framework". In [36] it demonstrates mechanism to pertain "MapReduce" on "Hadoop" to large-scale in "RDFS" inference.

The "Grids" and "Clouds" are recent appearances of distributed computing, placing a high value on "virtualization" and "software services technologies". The grid is synchronize resource contribution and difficulty-outcomes in a "dynamic, multi-agency virtual organization" [30]. The cloud can be deployed and capitalized with minimal administrative effort, and little is known about the infrastructure that supports it. This part temporarily introduces the function of policy and regulation interpretation in improving grid resources and workflow management. Generally of these outcome also pertain to the cloud computing environment. In [9] author introduced a innovative method for accessible performance in on-demand synthesis grids with pertain "ontology rules". The "Rule-based synthesis" unites multiple original actions into innovative composite actions. In [12] described "WS-CAM", a collaborative perceptual management function in a "rules-based grid" environment.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Finally, the "rule-based approaches" prove to be constructive for employ flexible manage schemes and conclusions that permits grids to use "service level agreement (SLA) management systems" to achieve the quality of service promised by various applications.

D. P2P computing for Rule-based Systems

In a "Peer-to-peer (P2P)" representation of distributed systems in which it "equally weighted" and "directly connected peers" work together through given that sources and services to every former peer. The "P2P systems" have significant purposes in "distributed processing", "distributed content management", and "ubiquitous computing". The grouping of "decentralization of the P2P approach" and the declaration and elasticity of the regulations facilitates the improvement of innovative category of intellectual distributed schemes. Applications are provided in heterogeneous schema mapping and ubiquitous computing.

In [34] author introduces the use of an inference engine to represent and process the meaning of digital library resources in a heterogeneous environment. This approach applies to metadata mapping between heterogeneous schemas in a "P2P-based digital library". The Mappings are characterize by taken out facts from the resource's "XML metadata" and then pertaining the "rule-based reasoning" to mechanically extract relationships among the confined schema and erstwhile discovered representations.

In [22] commences a "Prolog's P2P extension", "LogicPeer". In the "LogicPeer" it is a simple addition of "Prolog" using an operator that can perform target assessment on peers in a "P2P system". In "LogicPeer" it describes two network representations. (i) an "opaque peer network model" where every peer does not recognize the identifier of the neighbor router and the specific "query propagation protocol", (ii) each peer gets its neighbor's identifier, thus allowing the implementation of a customized query propagation protocol.

In [25] and [27] a distributed interpretation resolution is proposed that can utilized as a "P2P network environment". Each agent has a partial environmental perspective and has a locally consistent theory. Local theory is linked through bridging rules, which can lead to inconsistencies in the global knowledge base. Handling inconsistencies are done by creating a bridging rule with unmanageable logic.

V. Investigation Of The Related Works

Various researchers have keen to the significance of failing data analysis and the necessitate for an open failed data store. In this white paper, many failed data collected over the last decade on HPC sites have been studied and made publicly available [1].

J. Villamayor et al. [1] presents the "Fault Tolerance Manager (FTM)" for the coordinated checkpoint file to provide automatic recovery from failure when losing the coordinate node. This proposal makes the "Fault Tolerance (FT)" configuration simpler and more transparent for users without knowledge of application implementation. In addition, system administrators do not need to install libraries in the cluster to support FTM. Leverages node local storage to store checkpoints and distributes copies of nodes along all compute nodes to prevent centralized storage bottlenecks. This approach is especially useful in IaaS cloud environments where users have to pay for centralized, reliable storage services. This is based on "RADIC", a well-known architecture that provides FT in a distributed, flexible, and automatically scalable manner. Experimental results demonstrate the benefits of the approach presented in "Amazon EC2", a private cloud and well-known cloud computing environment.

Saurabh Jha et al. [2] has featured Cray's largest Gemini network Gemini Internet recovery procedure featured at "Blue Rock, a 13.3 petaflops supercomputer" at the "National Center for Computational Applications (NCSA)". It presents a propagation model that captures the failure and recovery of interconnects to help understand the types of faults and their propagation in systems and applications during recovery. The measurement results show that the recovery process is very frequent and the recovery process is unsuccessful in the incident of an additional failure during recovery, resulting in system-wide interruptions and application failures.

I. A. Sidorov [3] focuses on the development of models, methods, and tools to improve the FT of HPC-S. The models and methods described are based on the use of automated diagnostics, automatic localization, error correction, and automatic HPC-S reconfiguration mechanisms for the underlying software and HW components of these systems. The uniqueness and novelty of the proposed approach allow agents to make decisions directly by creating a multi-agent system using a general-purpose software agent that can collect node state data for analysis.

S. Hukerikar et al. [4] proposes a partial memory protection scheme based on area-based memory management. This defines a local concept called havens, which provides error protection for program objects. It also provides reliability for the area through a software-based parity protection mechanism. U. Lakhina et al. [5] The centralized load-balancing algorithm proposed an algorithm that dynamically balances the load and ensures the overall performance of the system. This concept focuses on building FT systems by achieving high resource utilization, reducing job rejection rates, and making improved calculations and backups. The results of this cluster-oriented load-balancing algorithm show that response time is reduced and processing time is improved.

We encourage this data to be the first action toward a open data store and support former sites to accumulate and organize data for open disclosure. Underneath we review some outcomes.

- Failure rates vary widely from system to system, and there are more than 20 to 1000 occurrences each year, depending on system dimension and HW type.
- Since the error rate is approximately relative to the quantity of processors in the system, the error rate does not increase much faster in proportion to the system dimension.
- There is confirmation that there is a relationship among the breakdown rate of the system and the kind and strength of

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

the executing jobs load. This is consistent among the initial job on another type of system.

- The curve of the breakdown rate above the life span of the HPC-S is frequently extremely dissimilar from the life cycle curve details in the background study of individual HW or SW modules.
- The occasion among failures is not well created through an "exponential distribution" that matches the initial discovery for other types of systems. Identify the time to failure of the entire system as well as individual nodes.
- The average repair instance fluctuate from "one to more than one hour", depending on the system. The repair time is largely dependent on the type of system and is not affected by the size of the system.
- Restore occasions are highly inconsistent inside a system and are a large amount of enhanced formed through logarithmic normal rather than the "exponential distribution".

This survey task has difficulties in solving key challenges and problems that can be improved to provide new solutions in the field of HPC-S. The key enhancements should focus on the FT framework of HPC [3], [9] that able to utilized to construct consistent HPC-S using rule-based error prediction methods. Investigate the impact of various parameters on overall forecast results and highlight the best combination of the log file and the described incidents of failure. Even the implementation of a hybrid FT approach that combines fault prediction and multi-step checkpoint techniques can overcome system overhead for many HPC applications [1], [6], [8], [19].

VI. Conclusion

High-performance computing (HPC) systems can use interconnection networks, allowing applications to execute continuously, regardless of size. In the incident of a failure, the interconnect recovery mechanism coordinates complex operations to recover network connectivity between nodes. As the size and design complexity of HPC-S increases, the system may fail during the interconnect recovery procedure. This review focuses on insights into HPC components and continues to grow to meet performance and user needs for science and other applications. It focuses on the importance and FT mechanisms and causes of HPC. Later, we discuss related works related to the function of HPC in "rule-based distribution systems". Understanding the challenges of HPC and the research support that can be improved in the future are needed to reduce mean time to recovery and propose new fault tolerance solutions in these systems to increase high availability.

References

[1] J. Villamayor, D. Rexachs, E. Luque, "A Fault Tolerance Manager with Distributed Coordinated Checkpoints for Automatic Recovery", International Conference on High Performance Computing & Simulation (HPCS), pp. 452 - 459, 2017.

[2] S. Jha, V. Formicola, C. Di Martino, M. Dalton, et al., "Resiliency of HPC Interconnects: A Case Study of Interconnect Failures and Recovery in Blue Waters", IEEE Transactions on Dependable and Secure Computing, Volume: PP, Issue: 99 pp. 1 - 1, 2017.

[3] I. A. Sidorov, "Methods and tools to increase fault tolerance of high-performance computing systems", 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 226 - 230, 2016.

[4] S. Hukerikar, C. Engelmann, "Havens: Explicit reliable memory regions for HPC applications", IEEE High Performance Extreme Computing Conference (HPEC), pp. 1 - 6, 2016

[5] U. Lakhina, N. Singh, A. Jangra, "An efficient load balancing algorithm for cloud computing using dynamic cluster mechanism" 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 1799 - 1804, 2016.

[6] I. Gankevich, Y, Tipikin, V. Korkhov, V. Gaiduchok, "Factory: Non-stop batch jobs without checkpointing", International Conference on High Performance Computing & Simulation (HPCS), pp. 979 - 984, 2016.

[7] D. W. Kim, M. Erez, "Balancing reliability, cost, and performance tradeoffs with FreeFault", IEEE 21st International Symposium on High Performance Computer Architecture (HPCA) pp. 439 - 450, 2015.

[8] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems", IEEE Transactions on Dependable and Secure Computing, vol. 7, no. 4, pp. 337-350, 2010.

[9] I. P. Egwutuoha, S. Chen, D. Levy, and B. Selic, "A Fault Tolerance Framework for High Performance Computing in Cloud", in 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 709-710, IEEE Press, 2012.

[10] R. Riesen, K. Ferreira, and J. Stearley, "See applications run and throughput jump: The case for redundant computing in HPC", in 2010 International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 29-34, IEEE Press, 2010.

[11] I. P. Egwutuoha, D. Levy, B. Selic, and S. Chen, "A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems", in The Journal of Supercomputing, vol. 65, pp. 1302-1326, Springer, 2013.

[12] N. Yigitbasi, M. Gallet, D. Kondo, A. Iosup, and D. Epema, "Analysis and modeling of time-correlated failures in large-scale distributed systems", in 2010 11th IEEE/ACM International Conference on Grid Computing (GRID), pp. 65-72, IEEE Press, 2010.

[13] N. Xiong, M. Cao, J. He and L. Shu, "A Survey on Fault Tolerance in Distributed Network Systems", Proceedings of the 2009 International Conference on Computational Science and Engineering, Vancouver, 29-31,1065-1070, 2009.

[14] E. N. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems", ACM Computing Surveys (CSUR), vol. 34, no. 3, pp. 375-408, 2002.

[15] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, et al., "A view of cloud computing", Communications of the ACM, vol. 53, no. 4, pp. 50-58, 2010.

[16] C. Evangelinos and C. Hill, "Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazon EC2", ratio, vol. 2, no. 2.40, pp. 2-34, 2008.

[17] Amazon EC2, "Amazon Elastic Compute Cloud (Amazon EC2)." http: //aws.amazon.com/ec2/. Online: accessed in April, 2013.

[18] A. Geist and C. Engelmann, "Development of naturally fault tolerant algorithms for computing on 100,000 processors", Submitted to Journal of Parallel and Distributed Computing, 2002.

[19] J. C. Sancho, F. Petrini, K. Davis, R. Gioiosa, and S. Jiang, "Current practice and a direction forward in checkpoint/restart implementations for fault tolerance", in 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), p. 300.2, IEEE Press, 2005.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

[20] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir, "Toward exascale resilience", International Journal of High Performance Computing Applications, vol. 23, no. 4, pp. 374-388, 2009.

[21] F. Cappello, "Fault tolerance in petascale/exascale systems: Current knowledge, challenges and research opportunities", International Journal of High Performance Computing Applications, vol. 23, no. 3, pp. 212-226, 2009.

[22] H. W, Meuer, and others., "TOP500 Supercomputer Sites." http://www.top500.org/. Online: accessed in April, 2013.

[23] Wikipedia.org, "Fault-tolerant system." http://en.wikipedia.org/wiki/Fault_tolerant_system. Online: accessed in April, 2013.

[24] USENIX, "Computer failure data repository (cfdr)." https://www. usenix. org/cfdr. Online: accessed in April, 2013.

[25] Arif Sari, Murat Akkaya, "Fault Tolerance Mechanisms in Distributed Systems", Int. J. Communications, Network and System Sciences, Vol. 8, PP. 471-482, 2015

[26] C. D. Martino, S. Jha, W. Kramer, Z. Kalbarczyk, and R. K. Iyer, "Logdiver: a tool for measuring resilience of extreme-scale systems and applications", in Proceedings of the 5th Workshop on Fault Tolerance for HPC at eXtreme Scale, pp. 11-18, ACM, 2015.

[27] J. B. Leners, H. Wu, W.-L. Hung, M. K. Aguilera, and M. Walfish, "Detecting failures in distributed systems with the falcon spy network", in Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, pp. 279-294, ACM, 2011.

[28] N. El-Sayed and B. Schroeder, "Reading between the lines of failure logs: Understanding how HPC systems fail", in Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on, pp. 1-12, IEEE, 2013.

[29] Los Alamos National Laboratory., "Operational Data to Support and Enable Computer Science Research". http://institute.lanl.gov/data/fdata/, 2014. Online: accessed in June, 2014.

[30] M. Nagappan, A. Peeler, and M. Vouk, "Modeling cloud failure data: a case study of the virtual computing lab", in Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing, pp. 8-14, ACM, 2011.

[31] C. Di Martino, F. Baccanico, J. Fullop, W. Kramer, Z. Kalbarczyk, and R. Iyer, "Lessons learned from the analysis of system failures at petascale: The case of blue waters", in Proc. of 44th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN), 2014.

[32] V. Puente, J. A. Gregorio, F. Vallejo, and R. Beivide, "Immunet: A cheap and robust fault-tolerant packet routing mechanism", in Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on, pp. 198-209, IEEE, 2004.

[33] D. Oppenheimer and D. A. Patterson, "Architecture and dependability of large-scale internet services", IEEE Internet Computing, vol. 6, no. 5, pp. 41-49, 2002.

[34] M. Balazinska, H. Balakrishnan, S. Madden and M. Stonebraker, "Fault-Tolerance in the Borealis Distributed Stream Processing System. ACM Transactions on Database Systems, 33, 1-44, 2008.

[35] E.N.M. Elnozahy, L. Alvisi, Y.M. Wang and D.B. Johnson, "A Survey of Rollback-Recovery Protocols in Message-Passing Systems. ACM Computing Surveys, 34, 375-408, 2002.

[36] D. Tian, K. Wu and X. Li, "A Novel Adaptive Failure Detector for Distributed Systems", Proceedings of the International Conference on Networking, Architecture, and Storage, Chongqing, PP. 215-221, 2008.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500